APPENDIX

```
.MODULE/ABS=0 LX1;
.pagelength 44;
/* This software is written in the native assembly language
of the Analog Devices 2181 Digital Signal Processor.
The purpose of this code is to oversample bead flu  rescence
in each of 5 consecutive channels t   isolate single bead events
and report the fluor area in each channel. The program
double buffers the data for the 167 micro, and uses the
following interrupts for interprocessor signalling
From 167 to IRQ this program:
p2.0__IRQ0        go to idle state
p2.1__IRQ1        new data in control area
p2.2__FI                  begin capture (this is a state, not IRQ)
p2.3__PWRDN nmi           begin execution at PM 0x2c

From this program to IRQ 167:
FL0__EX0IN        buffer 0 is ready
FL1__EX1IN        buffer 1 is ready
FL2__EX2IN        error occurred during capture state

All PM and DM are accessible from the 167 thru IDMA so
mem areas are asigned and restricted.
PM is loaded from EPROM but may be rewritten by:
1) IRQ0 (p2.0)
2) rewrite with vector to first instruction at 0x2c
3) PWRDN nmi (p2.3)

DM allocation (in hex) is:
0-ff      reserved for 2181
100-1ff   DSP status area, read only to 167
200-2ff   DSP control area, read all/ write restricted to 167
300-3ff   DSP command area, read/write 167, read / restricted write to DSP
400-9ff   buffer 0, 1536 words, 384 max block size
a00-fff   buffer 1
1000-3dff         reserved for DSP raw A/D data


*/


.CONST PFLAGS=0X3FE5;
/* DSPs variable area */
.VAR/dm/ABS = 0 version,actbuf,pki5,wid,svi6,svavg,svlft,fstart,lls,hls,svi3,svi7,svcry,haf,lo5,recbuf[5],left;
.VAR/dm/ABS = 0x80 bkgnd[10];
.VAR/dm/ABS = 0x90 inbkgnd[10];
.VAR/dm/ABS = 0xa0 x8bkgnd[10];
.VAR/dm/ABS = 0xb0 histbk[10];
/* DSP status */
.VAR/dm/ABS = 0X100 misscnt, xx1, buf0cnt, xx2, buf1cnt, xx3, otrcnt,xx4,afreg,xx5,nobgsub;
/* DSP control */
.VAR/dm/ABS = 0X200 buf0siz, xa1, buf1siz, xa2, thresh, xa3, mcready, xa4, clock, xa5,
lena,xa6,bcntl,xa7,bcnth;
/* DSP command */
.VAR/dm/ABS = 0X300 address,xxb1,hiword,xxb2,lobyte,xxb3,msg;
.VAR/dm/ABS = 0X400 outbuf0;

.VAR/dm/ABS = 0Xa00 outbuf1;
/* raw data area */
.VAR/dm/abs = 0x1000 hist[0x1000] ;

.VAR/dm/circ/abs = 0x2000 workbuf[0x1e00] ;
```

```
        /* interrupt vectors */

        dis ints;
        jump begin;      /*reset irq*/
        nop;
        nop;

        jump irqfl;      /*read 4 fluorescence measurements*/
        nop;
        nop;
        nop;

        rti;             /*irqL1 */
        nop;
        nop;
        nop;

        rti;             /*irqL0 */
        nop;
        nop;
        nop;

        rti;             /*sp0 tx*/
        nop;
        nop;
        nop;

        rti;             /*sp0 rx*/
        nop;
        nop;
        nop;

        rti;             /*irq E */
        nop;
        nop;
        nop;

        jump bdmadone;           /*irq BdmA*/
        nop;
        nop;
        nop;

        jump procnewctl;         /*irq1-new data in control area*/
        nop;
        nop;
        nop;

        jump doidle;     /*go to idle mode, wait for nmi*/
        nop;
        nop;
        nop;

        jump tim_exp;      /*timer*/
        nop;
        nop;
        nop;

softload:
        m0=0;
        l0=0;
        i0=^address;
```

```
          ax0=-1;
          dm(i0,m0)=ax0;
          l1=0;
          l2=0;
          l5=0;
          ax0=0xd000;       /* set up restart params */
          dm(0x3fef)=ax0;
          i1=^hiword;
          i2=^lobyte;
ldpgm:    ay0=dm(i0,m0);
          ar=pass ay0;
          if lt jump ldpgm;  /* negative number means no new program yet */
          i5=ay0;            /* point to addr of new instruction */
          ax0=dm(i2,m0);     /* get low byte */
          px=ax0;
          ax0=dm(i1,m0);     /* get hi word */
          pm(i5,m5)=ax0;     /*m reg doesn't matter */
          ax0=-1;
          dm(i0,m0)=ax0;
          ar=pass ay0;       /* if 0 then its the last instruction */
          if eq jump ldend;
          set fl2;
          nop;
          reset fl2;
          jump ldpgm;
ldend:    idle ;
          nop;
          nop;
          nop;
          nop;
          nop;
          nop;
          nop;
          nop;
          nop;
          nop;
          nop;
          nop;
          nop;
          nop;
          nop;
          nop;
          nop;
          nop;
          nop;
          nop;
          nop;
          nop;
          nop;

          /* initalization sequence */

begin:    imask=0;
          icntl=0x7;     /*no nesting, irq2,1,0=edge*/
          dis timer;
          l0=0;
          l1=0;
          l2=0;
          l3=0;
          l7=0;
          l6=0x1e00;
          mstat=0x10;              /*integer placement*/
          ax0=0;
```

```
        dm(lena)=ax0;
        m0=2;
        m1=-1;
        m2=1;
        m3=0;
        m4=-20;
        m5=1;
        m6=5;
        m7=0;
        i1=0x3fff;
        dm(i1,m1)=0;
        dm(i1,m1)=0x3ff8;          /* atod is ZERO wait state */
        dm(i1,m1)=0;      /*tperiod*/
        dm(i1,m1)=0;      /*tcount*/
        dm(i1,m1)=15;        /*tscale to 1 us per cnt*/
        dm(i1,m1)=0;        /*sp0*/
        dm(i1,m1)=0;
        dm(i1,m1)=0;
        dm(i1,m1)=0;
        dm(i1,m1)=0;        /*3ff6*/
        dm(i1,m1)=0;
        dm(i1,m1)=0;
        dm(i1,m1)=0;
        dm(i1,m1)=0x4000;                /*ena internal clock gen*/
        dm(i1,m1)=15;              /*ser clock out to 1.024 MHz*/
        dm(i1,m1)=0;               /* addr 3ff0*/
        dm(i1,m1)=0xd000;                /* set up power down options */
        i1 = 0x3fe5;
        dm(i1,m2)=0x0;      /*set pf0-7 to 0 */
        dm(i1,m1)=0x78ff;    /*pf0-7 are outputs*/
        reset fl0;
        reset fl1;
        reset fl2;
        reset flag_out;                  /* set flags for reg control */
        ax0=0;
        dm(msg)=ax0;
        dm(mcready)=ax0;
        dm(bcntl)=ax0;
        dm(bcnth)=ax0;
        dm(nobgsub)=ax0;


        cntr=0xc00;
        i0=^outbuf0;
        do clrmem until ce;
clrmem:             dm(i0,m2)=ax0;
        l4=0x1e00;
        i4=^workbuf;




/* let system stabilize */

        cntr=1000;
        do bpirq until ce;
        cntr=16000;
        do bpirq1 until ce;
bpirq1: nop;
bpirq:  nop;
```

```
        my1=i  (0);        /* init for first readapds */
        i1 = 0x3f  5;
        dm(i1,m2)=0x0;        /*set pf0-7 to 0 */

        call lightsoff;
        ifc=0xff;          /*clear all pending irqs*/

        jump mainbody;  /* for testing only */

        jump waitmc;      /* wait for micro to init data areas */


        /* INTERRUPT SERVICE ROUTINES-- */


   irqfl:
        my1=io(1);         /* read 0 setup 1 */
        dm(i4,m5)=my1;
        my1=io(2);         /* read 1 setup 2 */
        dm(i4,m5)=my1;
        my1=io(3);         /* read 2 setup 3 */

        /* ************************************* for new board, change to io(4) below */
        /* ************************************* for old board, change to io(3) below */
        dm(i4,m5)=my1;
        my1=io(4);         /* read 3 setup 4 */
        dm(i4,m5)=my1;
        my1=io(0);         /* read 4 setup 0 */
        dm(i4,m5)=my1, af=af+1;


        rti;


  procnewctl:
        rti;

  doidle:
        rti;


  bdmadone:
        rti;


  tim_exp:
        rti;




  /*      This loop is executed while NOT capturing. It only responds to 2 interrupts
          IRQ 1 means that there is new data in the control area so reinit all variables
          IRQ 0 means that there is new code in dm that must be moved to pm and executed

   */

  Mainbody:
          if flag_in call capture;
          ax0=dm(lena);     /*turn on lasers */
          ar=setbit 7 of ax0; /*enable overflow bit of a/d converter */
          dm(pflags)=ar;
        imask = 0x206;   /*allow idle or n  w ctl msg */
        ena ints;
        nop;
        nop;
        nop;
        nop;
        i0=^workbuf;
        ax0=dm(i0,m2);  /* write background in same order as recbuf */
        dm(bkgnd)=ax0;

        ax0=dm(i0,m2);
        dm(bkgnd+2)=ax0;

        ax0=dm(i0,m2);
        dm(bkgnd+4)=ax0;

        ax0=dm(i0,m2);
        dm(bkgnd+6)=ax0;

        ax0=dm(i0,m2);
        dm(bkgnd+8)=ax0;


        jump mainbody;





        /* All subroutines here at the end  */



capture:
        I4=0x1e00;
        I5=0x1e00;
        ax0=dm(lena);    /*turn on lasers */
        ar=clrbit 7 of ax0;          /* disable overflow bit of a/d convt */
        dm(pflags)=ar;
        dis ints;
        call clrhist;
        af=pass 0;
        i4=^workbuf;
        i3=^hist;
        i5=^workbuf;
        i7=^x8bkgnd;
        imask=0x200;
        ena ints;

        cntr=4;  /* do four background histograms */
        do bigloop until ce;

        cntr=4000;         /* take 4000 samples for histogram */
        do bldhist until ce;
pace:   ar=pass af;        /* any samples to proc */
        if le jump pace;
        ax0=dm(i5,m6);  /* get samp */
        ar=ax0-0x1000;  /* range check */
        if ge jump bldhist;
        ay0=^hist;         /* base addr */
```

```
           ar=ax0+ay0;      /* base + offset */
           i3=ar;           /*point into hist */
           ay1=dm(i3,m3);   /* get current val */
           ar=ay1+1;
           dm(i3,m3)=ar;
  bldhist: nop;

           /* now find most frequent occurrance in hist */

           i3=^hist;
           cntr=0xfff;
           ax0=dm(i3,m2);  /* get first entry */
           dm(svi3)=i3;
           ay0=ax0;                    /* ay0 holds highest val so far */

           do findmax until ce;
           ax0=dm(i3,m2);
           ar=ax0-ay0;
           if le jump findmax;
           ay0=ax0;                    /*new high val found */
           dm(svi3)=i3;       /* save offset +1 (post inc) */
 findmax:            nop;
           ax0=dm(svi3);      /* get ptr to best +1 */
           ay0=^hist + 1;     /* since i3 is plus 1 */
           ar=ax0-ay0;        /*compensate for base */
           si=ar;             /* save bkgnd for shift later */
           ay0=^hist;
           ar=ar+ay0;         /*calc index to peak */

           i3=ar;             /*point to peak in hist */
           ay0=dm(i3,m1);     /*get pk, pnt to previous */
           ax0=dm(i3,m0);     /*get prev, pnt to other side of pk */
           ay1=dm(i3,m1);     /* pnt back to peak */
           ax1=1;
           ar=ax0-ay1;        /* which pk is second highest */
           if lt jump poside;
           ax1=-1;            /* int pk will be less */
           ay1=ax0;

 poside:                      /* now ay1 has val of 2nd peak */
          sr=lshift si by 3 (lo);        /* mult by 8 */
          si=ay0;
          ay0=sr0;                       /*ay0 is 8x bkgnd and is ready for adj */
          sr=lshift si by -2(lo);        /* get 1/4 of peak height */
 intlp:   ar=ay1-sr0;        /* reduce 2nd pk by 1/4 of main */
          if lt jump intend;
          ay1=ar;
          ar=ax1+ay0;        /*adjust background + or - */
          ay0=ar;
          jump intlp;
 intend:  dm(i7,m5)=ay0;  /* save final bkgnd */

          call clrhist;
 bigloop:           modify(i5,m5);    /*point to next channel */

          i3=^x8bkgnd;
          i1=^bkgnd;
          cntr=4;
          do div8 until ce;
          si=dm(i3,m2);      /*get avg background , inc +2*/
          sr=lshift si by -3 (lo);
 div8:    dm(i1,m0)=sr0;
```

```
          af=pass 0;
          ax0=0;
          dm(misscnt)=ax0;
          dm(buf0cnt)=ax0;
          dm(buf1cnt)=ax0;
          dm(otrcnt)=ax0;
          dm(actbuf)=ax0;
          ax0=dm(clock);
          dm(0x3ff1)=ax0;  /* set sample clk freq */
          i4=^workbuf;
          i5=^workbuf;
          ifc=0xff;
          imask=0x200;     /* ena fl ints */
          i0=^outbuf0;
resetc:   if not flag_in jump capexit;
          ax0=0;
          ar=pass af;
          dm(afreg)=ar;
          ena ints;
newsamp:            if not flag_in jump capexit;
          ar=pass af;                    /* load ax0 with # of new samps */
          ar=ar-64;          /* are we 64 records  ahead */
          if le jump newsamp;
          af=af-1,ay0=dm(i5,m6); /*get ssc, inc to next record */
          ax0=dm(thresh);
          ar=ax0-ay0;        /* above thresh then its a bead */
          if ge jump newsamp;

          /* got a bead */

          dm(pki5)=i5;
          dm(fstart)=i5;
          ax1=ay0;                       /* init peak ssc */
          ay1=0;  /* init width count */

moresamp:
          af=af-1,ay0=dm(i5,m6); /* next next ssc, dec rec count */
          ar=ay1+1;                      /* inc width count */
          ay1=ar,ar=ax1-ay0 ;            /* is new ssc higher */
          if ge jump ckdone;
          dm(pki5)=i5;
          ax1=ay0;                       /* save peak ssc */
ckdone:             ar=pass af;          /* out of records */
          if eq jump bubble;             /* if 0 then bubble */
          ar=ax0-ay0;        /* back below thresh yet? */
          if le jump moresamp;
          i6=dm(pki5);
          modify(i6,m4);     /* since post inc, sub 20 */
          dm(wid)=ay1;

          /* build a record with peaks or estimates of peaks */
          l2=0x1e00;
          i2=i6;
          call i2sum;
          dm(recbuf)=ay1;  /*store ssc in recbuf*/

          modify(i6,m5);     /* add 1 to point to fl2 pk -15 */
          i2=i6;
          call i2sum;
          dm(recbuf+1)=ay1;              /* store cl1 sum */
          modify(i6,m5);    /* add 1 to point t  fl3 pk -15 */
          i2=i6;
          call i2sum;
          dm(recbuf+2)=ay1;         /* store cl2 sum */

          modify(i6,m5);    /* add 1 to point to fl4 pk -15 */
          i2=i6;
          call i2sum;
          dm(recbuf+3)=ay1;         /* store cl3 sum */

          modify(i6,m5);    /* add 1 to point to fl1 pk -15 */
          dm(svi6)=i6;      /*align point with red laser */
          modify(i6,m4);    /* back up 20 more to fl1 peak -35 */
          modify(i6,m4);    /* back up 20 more to fl1 peak -55 */
          /*modify(i6,m4);   back up 20 more to fl1 peak -75 */

          /* since fl1 is a different laser, refind peak to allow for alignment
          differences, search from -15 to +16 */

          ax0=dm(i6,m6);  /* get 1st fl1 sample */
          m7=160;
          m7=350;                   /*for new filter at 50 khz */
          modify(i6,m7);
          m7=0;
          ay0=dm(i6,m6);
          ar=ax0+ay0;
          sr=lshift ar by -2 (LO);
          ay0=sr0;
          ar=ar+ay0;
          dm(x8bkgnd+4)=ar;


          i6=dm(svi6);      /* test using no peak find */
          m7=40;
          modify(i6,m7);    /*move right 8 samples due to skew of 50k filter */
          m7=0;

          cntr=7;
          modify(i6,m6);
          dm(pki5)=i6;
          ay0=dm(i6,m6);  /* get 1st fl1 sample */
          ax1=ay0;                  /* move to old fl1 val */
          do rp1pk until ce;
          ay0=dm(i6,m6);
          ar=ax1-ay0;       /* is it a new peak?*/
          if gt jump rp1pk;
          dm(pki5)=i6;      /* store addr +5 of new peak (post inc) */
          ax1=ay0;
rp1pk:    nop;
          i6=dm(pki5);
          modify(i6,m4);    /* sub 20 to back up 3 samples from pk (was +5)*/
          modify(i6,m4);    /* sub 20 to back up 4 samples 50 k version*/
          i2=i6;
          call i2sum14;
          sr=lshift si by -3 (HI);       /* shift carry */
          sr=sr or lshift ar by -3 (LO);
          dm(recbuf+4)=sr0;        /*store rp1 */
          l2=0;             /* restore for non circ buffr use */
          /* ok, we have singlet and recbuf is built */

          i7=^recbuf;
          i3=^x8bkgnd;
          cntr=4;
```

```
            do subback until ce;
            ax0=dm(i7,m7);   /* get val inc 0 */
            ay0=dm(i3,m2);
            ar=ax0-ay0;
            if lt ar=pass 0;
  subback:           dm(i7,m5)=ar;
            ax0=dm(nobgsub);
            ar=pass ax0;
            if ne jump recbdone;
            ax0=dm(i7,m7);   /* get val inc 0 */
            ay0=dm(i3,m2);
            ar=ax0-ay0;
            if lt ar=pass 0;
            dm(i7,m5)=ar;
            /*********Sub r/t bg from rp1*********/


  recbdone:

                    /* record bead count */
           ay1=dm(bcntl);
           ar=ay1+1;
           dm(bcntl)=ar;
           ar=dm(bcnth);
           ar=ar+C;
           dm(bcnth)=ar;


           i2=^recbuf;
           ay1=dm(actbuf);
           none=pass ay1;
           if ne jump usebuf1;

           ay1=dm(buf0cnt);
           none=pass ay1;
           if gt jump noi0init;          /* is this first of block */
           i0=^outbuf0;
 noi0init:
           ax1=dm(buf0siz);              /* check for overflow */
           none = ax1 - ay1;
           if ne jump noover;
           ay1=dm(misscnt);
           ar=ay1+1;
           dm(misscnt)=ar;
           jump resetc;
 noover:   cntr=5;
           do movout0 until ce;
           ar=dm(i2,m2);
 movout0:           dm(i0,m0)=ar;    /* store ssc inc by 2 for even addr*/


          ay0=dm(buf0cnt);
          ar=ay0+1;
          dm(buf0cnt)=ar;
          ay0=dm(buf0siz);               /* enough records to irq 167? */
          ar=ar-ay0;
          if ne jump resetc;
          set fl0;
          nop;                  /* c167 needs 50 ns min to get edge */
          nop;
          reset fl0;
          ay1=1;
          dm(actbuf)=ay1;  /* change to buffer 1 */
```

```
        jump resetc;


usebuf1:
        ay1=dm(buf1cnt);
        none=pass ay1;
        if gt jump noi1init;            /* is this first of block */
        i0=^outbuf1;
noi1init:
        ax1=dm(buf1siz);                /* check for overflow */
        none = ax1 - ay1;
        if ne jump noover1;
        ay1=dm(misscnt);
        ar=ay1+1;
        dm(misscnt)=ar;
        jump capture;       /* major overflow reset and restart capture */
noover1:        cntr=5;
        do movout1 until ce;
        ar=dm(i2,m2);
movout1:        dm(i0,m0)=ar;     /* store ssc inc by 2 for even addr*/

        ay0=dm(buf1cnt);
        ar=ay0+1;
        dm(buf1cnt)=ar;
        ay0=dm(buf1siz);                /* enough records to irq 167? */
        ar=ar-ay0;
        if ne jump resetc;
        set fl1;
        nop;                    /* c167 needs 50 ns min to get edge */
        nop;
        reset fl1;
        ay1=0;
        dm(actbuf)=ay1;  /* change back to buffer 0 */
        jump resetc;

capexit:
        i1 = 0x3fe5;
        dm(i1,m2)=0x0;         /*set pf0-7 to 0 */

        ifc=0xff;
        rts;

i2sum14:
        m0=5;
        ar=pass 0,ay1=dm(i2,m0);
        si=0;          /* use for carry */
        dm(svcry)=si;
        cntr=20;
        do grate4 until ce;
        ar=ar+ay1,ay1=dm(i2,m0);
        if not ac jump grate4;
        dis ints;
        ena sec_reg;
        ay0=dm(svcry);
        ar=ay0+1;
        dm(svcry)=ar;
        dis sec_reg;
        ena ints;
grate4: nop;
        ay1=ar;
        si=dm(svcry);
        m0=2;
        rts;
```

```
i2sum:  m0=5;
        ar=pass 0,ay1=dm(i2,m0);
        cntr=8;
        do grate until ce;
grate:  ar=ar+ay1,ay1=dm(i2,m0);
        ay1=ar;
        m0=2;
        rts;

bubble:
        if not flag_in jump capexit;
        ar=pass af;                  /* load ax0 with # of new samps */
        ar=ar-256;          /* are we 64 records *4 ahead */
        if le jump bubble;
        af=af-1,ay0=dm(i5,m6); /*get ssc, inc to next record */
        ax0=dm(thresh);
        ar=ax0-ay0;         /* are we below thresh yet */
        if le jump bubble;           /* no, keep waiting */
        jump newsamp;


lightsoff:
        ax0=0x24d;
        ax0=0x63;           /******************test**********/
        dm(version)=ax0;
        rts;

clrhist:
        i3=^hist;
        ax0=0;
        cntr=0x1000;
        do zit until ce;
zit:    dm(i3,m2)=ax0;
        rts;

waitmc:
        toggle fo;
        ax0=dm(mcready);            /* wait until mc writes non zero to mcready */
        ar=pass ax0;
        if eq jump waitmc;
        jump mainbody;

.ENDMOD;
```